

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a pattern that resembles a stylized 'Y' or a complex geometric shape. The letters are black on a white background. The central vertical column is composed of 'Y's, while the horizontal and diagonal branches are composed of 'S's. The overall shape is a large, hollow 'Y' with a complex internal structure of 'S's and 'Y's.

```

LL               IIIIII   SSSSSSSS
LL               IIIIII   SSSSSSSS
LL               II       SS
LL               II       SS
LL               II       SS
LL               II       SS
LL               II       SSSSSS
LL               II       SSSSSS
LL               II       SS
LL               II       SS
LL               II       SS
LL               II       SS
LLLLLLLLLLLL    IIIIII   SSSSSSSS
LLLLLLLLLLLL    IIIIII   SSSSSSSS

```

ERRORLOG
Table of contents

- ERROR LOG SUPPORT ROUTINES

H 4

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00

Page 0

(1)	198	UNEXPECTED INTERRUPT SERVICE
(1)	315	LOG DEVICE ERRORS
(1)	407	LOG ASYNCHRONOUS DEVICE ATTENTIONS
(1)	492	LOG SOFTWARE STATUS
(1)	583	LOG DRIVER MESSAGE
(1)	644	ERL\$LOG DMSCP and ERL\$LOG TMSCP
(1)	703	BUILD STARTUP AND POWERFAIL MESSAGES
(1)	739	ALLOCATE ERROR MESSAGE BUFFER
(1)	803	GET FULL DEVICE NAME
(1)	844	RELEASE ERROR MESSAGE BUFFER
(1)	876	WAKE ERROR LOG FORMAT PROCESS

```
0000 1 .TITLE ERRORLOG - ERROR LOG SUPPORT ROUTINES
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27 D. N. CUTLER 7-MAR-77
0000 28
0000 29 ERROR LOG SUPPORT ROUTINES
0000 30
0000 31 MODIFIED BY:
0000 32
0000 33 V03-012 EAD0162 Elliott A. Drayton 26-Apr-1984
0000 34 Correct ADDB3 in routine GETFULLNAME to use R0.
0000 35
0000 36 V03-011 EAD0160 Elliott A. Drayton 16-Apr-1984
0000 37 Added a test for the system block address not being there.
0000 38
0000 39 V03-010 EAD0137 Elliott A. Drayton 11-Apr-1984
0000 40 Changed code to log full device names. NODE NAME + DEVICE.
0000 41
0000 42 V03-009 LMP0221 L. Mark Pilant, 30-Mar-1984 13:57
0000 43 Change UCBSL_OWNUIC to ORBSL_OWNER and UCBSW_VPROT to
0000 44 ORBSW_PROT.
0000 45
0000 46 V03-008 KPL0100 Peter Lieberwirth 22-Mar-1984
0000 47 Use CONFREGL instead of CONFREG. Anticipate SBICONF
0000 48 containing a PFN instead of a VA if BI adapter
0000 49 initialization didn't originally recognize the adapter.
0000 50
0000 51 V03-007 SSA0007 Stan Amway 2-feb-1984
0000 52 Fix broken branch to ERL$ALLOCEMB.
0000 53
0000 54 V03-006 LMP0185 L. Mark Pilant, 1-feb-1984 9:37
0000 55 Fix some broken branches.
0000 56
0000 57 V03-005 ROW0241 Ralph O. Weber 12-OCT-1983
```

0000	58 :	Correct ERL\$LOG_D(T)MSCP to allocate space for the error log
0000	59 :	header in addition to the space needed for the logged message
0000	60 :	etc. Also use symbolic size of error log entry header instead
0000	61 :	of a constant.
0000	62 :	
0000	63 :	V03-004 RLRMSCP Robert L. Rappaport 27-Jul-1983
0000	64 :	Add two entryptoints, ERL\$LOG_DMSCP and ERL\$LOG_TMSCP,
0000	65 :	to log invalid Disk MSCP and Tape MSCP messages.
0000	66 :	
0000	67 :	V03-003 KDM0051 Kathleen D. Morse 11-Jul-1983
0000	68 :	Change references to TODR to use loadable, cpu-dependent
0000	69 :	routine, EXESREAD TODR.
0000	70 :	V03-002 BLS0187 Benn Schreiber 24-Sep-1982
0000	71 :	Correct broken branch offset due to UCB growing
0000	72 :	
0000	73 :-	

```
0000 75 :  
0000 76 : MACRO LIBRARY CALLS  
0000 77 :  
0000 78 :  
0000 79 $CDDDBDEF ;DEFINE CDDB OFFSETS  
0000 80 $CDRPDEF ;DEFINE CDRP OFFSETS  
0000 81 $DDBDEF ;DEFINE DDB OFFSETS  
0000 82 $DDTDEF ;DEFINE DDT OFFSETS  
0000 83 $DEVDEF ;DEFINE DEVICE CHARACTERISTIC BITS  
0000 84 $EMBDEF <DV,SU,TS,UI,SP,LM,ET> ;ERROR LOG MESSAGE BUFFERS OFFSETS  
0000 85 $ERLDEF ;DEFINE ERROR ALLOCATION BUFFER OFFSETS  
0000 86 $FCBDEF ;DEFINE FCB OFFSETS  
0000 87 $IODEF ;DEFINE I/O FUNCTION VALUES  
0000 88 $IRPDEF ;DEFINE IRP OFFSETS  
0000 89 $MCHKDEF ;DEFINE MACHINE CHECK RECOVERY MASK BITS  
0000 90 $MSCPDEF ;DEFINE MSCP OFFSETS  
0000 91 $NDTDEF ;DEFINE NEXUS DEVICE TYPE CODES  
0000 92 $ORBDEF ;DEFINE OBJECT'S RIGHTS BLOCK OFFSETS  
0000 93 $PRDEF ;DEFINE PROCESSOR REGISTER NUMBERS  
0000 94 $SBDEF ;DEFINE SYSTEM BLOCK OFFSETS  
0000 95 $UCBDEF ;DEFINE UCB OFFSETS  
0000 96 $WCBDEF ;DEFINE WCB OFFSETS  
0000 97 :  
0000 98 :  
0000 99 ;DEBUG=1 ;***  
0000 100 ;***IF DEFINED, ENABLE UNEXPECTED  
0000 101 ;*** INTERRUPT IDENTIFIES VECTOR #  
0000 102 :  
0000 103 : LOCAL MACROS  
0000 104 :  
0000 105 :  
0000 106 :  
0000 107 : MACRO TO DEFINE AN INTERRUPT SERVICE ROUTINE LABEL FOR UNEXPECTED INTERRUPTS  
0000 108 :  
0000 109 .MACRO ISRDEF,VNUM  
0000 110 .ALIGN LONG ; Make all vectors long word alligned  
0000 111 ERL$VEC'VNUM:: ;INTERRUPT SERVICE LABEL  
0000 112 .IF DF,DEBUG ;***IF DEBUGGING  
0000 113 BSBW ERL$UNEXP ;***CALL INTERRUPT SERVICE  
0000 114 .BYTE <VNUM>/2 ;***IDENTIFY VECTOR OFFSET INTO SCB  
0000 115 .ENDC  
0000 116 .ENDM ISRDEF  
0000 117 :  
0000 118 : MACRO TO DEFINE THE INTERRUPT SERVICE ROUTINE LABELS FOR AN ADAPTER  
0000 119 :  
0000 120 .MACRO ADPISR,SLOT  
0000 121 VECTOR = SLOT * 4 + 256  
0000 122 .REPT 4  
0000 123 ISRDEF \VECTOR  
0000 124 VECTOR = VECTOR + <16 * 4>  
0000 125 .ENDR  
0000 126 .IF NDF,DEBUG ;IF NOT DEBUGGING  
0000 127 BSBB ADP_HANDLER ;CALL INTERRUPT SERVICE  
0000 128 .ENDC  
0000 129 .ENDM ADPISR  
0000 130 :  
0000 131 : LOCAL SYMBOLS
```

```
0000 132 :  
0000 133 :  
0000 134 :  
0000 135 : MAXIMUM NUMBER OF MESSAGES BEFORE WAKE OF FORMAT PROCESS  
0000 136 :  
0000 137 :  
0000000A 0000 138 MAXMSG=10 ;  
0000 139 :  
0000 140 :  
0000 141 : MAXIMUM TIME IN SECONDS BEFORE WAKE OF FORMAT PROCESS  
0000 142 :  
0000 143 :  
0000001E 0000 144 MAXTIM=30 ;  
0000 145 :  
0000 146 :  
0000 147 : LOCAL DATA  
0000 148 :  
0000 149 :  
00000000 0000 150 .PSECT $$$260,QUAD,WRT  
0000 151 :  
0000 152 : WARNING!!! The next two bytes must be adjacent and word aligned  
0000 153 :  
00 0000 154 .ALIGN WORD  
00 0000 155 BUF1: .BYTE 0 ;COUNT OF BUSY MESSAGES IN BUFFER  
00 0001 156 .BYTE 0 ;COUNT OF COMPLETED MESSAGES IN BUFFER  
00 0002 157 .BYTE 0 ;BUFFER INDICATOR  
00 0003 158 .BYTE 0 ;BUFFER CONTROL FLAGS  
0000000C 0004 159 .LONG 10$ ;ADDRESS OF NEXT AVAILABLE SPACE IN BUFFER  
00000200 0008 160 .LONG 20$ ;ADDRESS OF END OF BUFFER + 1  
00000200 000C 161 10$: .BLKB 512-ERL$C_LENGTH ;ACTUAL BUFFER AREA  
0200 162 20$: ;REF LABEL  
0200 163 :  
0200 164 : WARNING!!! The next two bytes must be adjacent and word aligned  
0200 165 :  
0200 166 .ALIGN WORD  
00 0200 167 BUF2: .BYTE 0 ;COUNT OF BUSY MESSAGES IN BUFFER  
00 0201 168 .BYTE 0 ;COUNT OF COMPLETED MESSAGES IN BUFFER  
01 0202 169 .BYTE 1 ;BUFFER INDICATOR  
00 0203 170 .BYTE 0 ;BUFFER CONTROL FLAGS  
0000020C 0204 171 .LONG 10$ ;ADDRESS OF NEXT AVAILABLE SPACE IN BUFFER  
00000400 0208 172 .LONG 20$ ;ADDRESS OF END OF BUFFER + 1  
00000400 020C 173 10$: .BLKB 512-ERL$C_LENGTH ;ACTUAL BUFFER AREA  
0400 174 20$: ;REF LABEL  
0400 175 :  
0400 176 :  
0400 177 : GLOBAL DATA  
0400 178 :  
0400 179 : ERROR LOG DATA BASE  
0400 180 :  
0400 181 :  
00000000 0400 182 ERL$AL_BUFADDR:: ;ALLOCATION BUFFER ADDRESS ARRAY  
00000200 0404 183 .LONG BUF1 ;ADDRESS OF BUFFER 1 DESCRIPTOR  
0408 184 .LONG BUF2 ;ADDRESS OF BUFFER 2 DESCRIPTOR  
00 0408 185 ERL$GB_BUFIND:: ;CURRENT ALLOCATION BUFFER INDICATOR  
00 0408 186 .BYTE 0 ;  
00 0409 187 ERL$GB_BUFFLAG:: ;BUFFER STATUS FLAGS  
00 0409 188 .BYTE 0 ;
```

ERRORLOG
V04-000

- ERROR LOG SUPPORT ROUTINES

M 4

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 5
(1)

	040A	189	ERL\$GB_BUFPTR::		:FORMAT PROCESS BUFFER INDICATOR
00	040A	190	.BYTE	0	:FORMAT PROCESS WAKEUP TIMER
	040B	191	ERL\$GB_BUFTIM::		:PROCESS ID OF ERROR LOG PROCESS
1E	040B	192	.BYTE	MAXTIM	:UNIVERSAL ERROR SEQUENCE NUMBER
	040C	193	ERL\$GL_ERLPID::		:
00000000	040C	194	.LONG	0	:
	0410	195	ERL\$GL_SEQUENCE::		:
00000000	0410	196	.LONG	0	:

```
0414 198 .SBTTL UNEXPECTED INTERRUPT SERVICE
0414 199
0414 200 :+
0414 201 : ERL$VEC'VNUM - INTERRUPT SERVICE FOR SCB VECTOR VNUM.
0414 202 : ERL$UNEXP - GENERAL UNEXPECTED INTERRUPT SERVICE
0414 203 :
0414 204 : THESE INTERRUPT SERVICE ROUTINES ARE EXECUTED FOR UNUSED SCB VECTORS.
0414 205 :
0414 206 : IF DEBUG IS DEFINED, EACH INTERRUPT SERVICE CALLS ERL$UNEXP WITH
0414 207 : THE <VECTOR OFFSET>/2 INTO THE SCB AS A 1 BYTE ARGUMENT.
0414 208 :
0414 209 : IF DEBUG IS NOT DEFINED, ALL CPU INTERRUPT SERVICE ROUTINES COLLAPSE TO
0414 210 : GLOBAL LABELS EQUAL TO ERL$UNEXP AND ALL ADAPTER INTERRUPT SERVICE
0414 211 : ROUTINES CALL A ROUTINE THAT SAVES THE ADAPTER TYPE, TRIES TO DISABLE
0414 212 : FURTHER INTERRUPTS, AND LOGS THE INTERRUPT.
0414 213 :
0414 214 : THERE ARE ENOUGH INTERRUPT SERVICE ROUTINES FOR THE ARCHITECTURAL PAGE
0414 215 : OF THE SCB, I.E., 128 ROUTINES.
0414 216 :
0414 217 : INPUTS:
0414 218 :
0414 219 : (SP) = PC AT INTERRUPT
0414 220 : 4(SP) = PSL AT INTERRUPT
0414 221 :
0414 222 : OUTPUTS:
0414 223 :
0414 224 : ERROR IS LOGGED, OR PROCESSOR BUGCHECKS.
0414 225 : -
00000000 226 : .PSECT $AEXENONPAGED, LONG
0000 227 :
0000 228 : UNEXPECTED ADAPTER INTERRUPT HANDLER: IF DEBUG IS DISABLED, SAVE THE
0000 229 : ADAPTER TYPE, ATTEMPT TO DISABLE FURTHER INTERRUPTS FROM THE ADAPTER,
0000 230 : AND LOG THE INTERRUPT. IF DEBUG IS ENABLED, BUGCHECK AS FOR CPU INTERRUPTS.
0000 231 :
0000 232 : .ALIGN LONG
0000 233 ADP_UNEXP:
0000 234 NEXUS = 0 ;FIRST ADAPTER = 0
0000 235 .REPT 16 ;ISR'S FOR 16 ADAPTERS ONLY
0000 236 ADPISR \NEXUS ;DEFINE ERL$INT'VNUM LABELS AND ISRS
0000 237 NEXUS = NEXUS + 1 ;NEXT ADAPTER
0000 238 .ENDR ;
003E 239 ADP_HANDLER:
003E 240 SUBL #ADP_UNEXP+2, (SP) ;COMPUTE ADAPTER OFFSET
0045 241 DIVL #4, (SP) ;COMPUTE ADAPTER SLOT/TR NUMBER
0048 242 PUSHR #*M<R0, R1, R2, R3, R4> ;SAVE REGISTERS
004A 243 MOVL 5*4(SP), R3 ;RETRIEVE SLOT NUMBER
004E 244 MOVL @MMG$GL_SBICONF[R3], R4 ;GET ADDRESS OF ADAPTER REGISTERS
0056 245 BGEQ 100$ ;GEQ MEANS SBICONF DOES NOT CONTAIN
0058 246 ; A SYSTEM VA, MUST BE PFN OR 0
0058 247
0058 248 $PRTCTINI B*5$, #<MCHK$M_NEXM!MCHK$M_LOG>
0064 249
0064 250 CLRL 4(R4) ;DISABLE ADAPTER INTERRUPTS (HOPEFULLY)
0067 251 MOVL (R4), R1 ;GET ADAPTER CONFIGURATION REG CONTENTS
006A 252 CMPB R1, #NDT$_DR32 ;IS THIS A DR32?
006D 253 BNEQ 1$ ;BRANCH IF NOT
006F 254 MOVL #*X500, (R4) ;ELSE CLEAR INTERRUPTS IN SPECIAL WAY
```

6E 00000002'8F C2 003E 240
6E 04 C6 0045 241
1F BB 0048 242
53 14 AE D0 004A 243
54 00000000'FF43 D0 004E 244
5C 18 0056 245
0058 246
0058 247
0058 248
0064 249
04 A4 D4 0064 250
51 64 D0 0067 251
30 51 91 006A 252
07 12 006D 253
64 00000500 8F D0 006F 254

```
54 64 D0 0076 255 1$: MOVL (R4),R4 ;GET THE ADAPTER'S CONFIGURATION REG
0079 256
0079 257 $PRTCTEND 5$
37 50 E9 007A 258 BLBC R0,100$ ;IF R0 LBC, THEN NO ADPATER PRESENT
007D 259
00000000'FF43 D5 007D 260 TSTL @EXE$GL_CONFREGL[R3] ;ALREADY CONFIGURED?
08 12 0084 261 BNEQ 10$ ;IF NEQ, YES
00000000'FF43 54 9A 0086 262 MOVZBL R4,@EXE$GL_CONFREGL[R3] ;SAVE THE ADAPTER TYPE
008E 263 10$:
51 18 D0 008E 264 MOVL #EMB$C_UI_LENGTH,R1 ;SET SIZE OF MESSAGE TO ALLOCATE
00000251'EF 16 0091 265 JSB ERL$ALLOCEMB ;ALLOCATE AN ERROR LOG BUFFER
14 50 E9 0097 266 BLBC R0,20$ ;BRANCH IF NONE AVAILABLE
04 A2 0061 8F B0 009A 267 MOVW #EMB$C_UI,EMB$W_UI_ENTRY(R2) ;SET MESSAGE TYPE
10 A2 53 D0 00A0 268 MOVL R3,EMB$C_UI_TR(R2) ;SET SLOT/TR NUMBER
14 A2 54 D0 00A4 269 MOVL R4,EMB$C_UI_CSR(R2) ;SET CONFIGURATION REGISTER VALUE
00000325'EF 16 00A8 270 JSB ERL$RELEASEMB ;RELEASE BUFFER
1F BA 00AE 271 20$: POPR #^M<R0,R1,R2,R3,R4> ;RESTORE REGISTERS
5E 04 C0 00B0 272 ADDL #4,SP ;REMOVE SLOT NUMBER
02 00B3 273 REI
00B4 274
54 D4 00B4 275 100$: CLRL R4 ;FLAG NO ADAPTER PRESETN
D6 11 00B6 276 BRB 10$ ;JOIN COMMON CODE
00B8 277
00B8 278 :
00B8 279 : UNEXPECTED CPU INTERRUPT HANDLER: IF DEBUG IS ENABLED, BUGCHECK WITH
00B8 280 : <VECTOR OFFSET>/2 INTO SCB AS TOP BYTE ON STACK. IF DEBUG IS DISABLED,
00B8 281 : JUST BUGCHECK.
00B8 282 :
00B8 283 : .ALIGN LONG
00B8 284 CPU_UNEXP:
00000000 00B8 285 VNUM=000 ;FIRST VECTOR = 0
00B8 286 .REPT 64 ;ISR'S FOR CPU INTERRUPTS ONLY
00B8 287 ISRDEF \VNUM ;DEFINE ERL$INT'VNUM LABEL AND ISR
00B8 288 VNUM=VNUM+4 ;NEXT VECTOR
00B8 289 .ENDR
00B8 290
00B8 291 ERL$UNEXP::
00B8 292 .IF DF,DEBUG ;***IF VECTOR ID ENABLED,***
00B8 293 MOVZBL @ (SP), (SP) ;***OVERLAY RETURN WITH VECTOR OFFSET
00B8 294 MULL #2, (SP) ;***CONVERT ARG TO VECTOR OFFSET
00B8 295 .IFTF
00B8 296 BUG CHECK UNXINTEXC ;BUGCHECK
00BC 297 .IFT
00BC 298 TSTL (SP)+ ;***CLEAN STACK
00BC 299 .ENDC
02 00BC 300 REI ;RETURN FROM INTERRUPT
00BD 301
00BD 302 :
00BD 303 : Vector entry for counting unexpected interrupts, rather than logging
00BD 304 : them. Used on 11/780 for passive release on the DW780 and for the
00BD 305 : CVTP microcode bug.
00BD 306 :
00BD 307 : .ALIGN LONG
00C0 308
00C0 309 ERL$VEC_RETURN::
00C0 310 INCL 10$GL_SCB_INT0 ; Increment counter
02 00C6 311 REI ; And return
```

ERRORLOG
V04-000

- ERROR LOG SUPPORT ROUTINES
UNEXPECTED INTERRUPT SERVICE

C 5

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 8
(1)

00C7 312
00C7 313

ERR
V04

```
00C7 315 .SBTTL LOG DEVICE ERRORS
00C7 316
00C7 317 :+ ERL$DEVICERR - LOG DEVICE CONTROLLER AND/OR DRIVE ERROR
00C7 318 ERL$DEVICTMO - LOG DEVICE TIMEOUT ERROR
00C7 319
00C7 320 THIS ROUTINE IS CALLED TO LOG A DEVICE TIMEOUT OR DEVICE CONTROLLER
00C7 321 AND/OR DRIVE ERROR.
00C7 322
00C7 323 INPUTS:
00C7 324
00C7 325 R5 = DEVICE UNIT UCB ADDRESS.
00C7 326
00C7 327 OUTPUTS:
00C7 328
00C7 329 IF AN ERROR LOG ENTRY IS NOT ALREADY IN PROGRESS ON THE UNIT, ERROR
00C7 330 LOGGING IS ENABLED FOR THE UNIT, AND THE CURRENT REQUEST DOES NOT
00C7 331 INHIBIT ERROR LOGGING, THEN AN ERROR MESSAGE BUFFER IS ALLOCATED AND
00C7 332 FILLED IN WITH PERTINENT REQUEST INFORMATION FOLLOWED BY A DUMP OF
00C7 333 THE DEVICE REGISTERS.
00C7 334
00C7 335 ALL REGISTERS ARE PRESERVED ACROSS CALL.
00C7 336
00C7 337
00000000 338 .PSECT WIONONPAGED
0000 339 .ENABL LSB
0000 340 ERL$DEVICERR:: :LOG DEVICE CONTROLLER AND/OR DRIVE ERROR
0000 341 PUSHL #EMB$C_DE :SET FOR DEVICE ERROR
0000 342 BRB 10$
0000 343 ERL$DEVICTMO:: :LOG DEVICE TIMEOUT ERROR
0000 344 MOVZWL #EMB$C_DT, -(SP) :SET FOR DEVICE TIMEOUT
0000 345 10$: BBS #DEV$V_EL$G,UCB$C_DEVCHAR(R5),15$ :IF SET, ERROR LOG ENABLED
0000 346 12$: BRW 40$ :ERROR LOG DISABLED
0000 347 15$: BBS #IOS$V_INHERLOG,UCB$C_FUNC(R5),12$ :IF SET, ERROR LOG INHIBITED
0000 348 INCW UCB$C_ERRCNT(R5) :INCREMENT NUMBER OF DEVICE ERRORS
0000 349 BBS #UCB$C_ERLOGIP,UCB$C_STS(R5),40$ :IF SET, ERROR IN PROGRESS
0000 350 PUSHR #M<R0,R1,R2,R3,R6> :SAVE REGISTERS
0000 351 MOVL UCB$C_DDB(R5),R3 :GET ADDRESS OF DDB
0000 352 MOVL UCB$C_DDT(R5),R6 :GET ADDRESS OF DDT (from UCB not DDB)
0000 353 MOVZWL DDT$C_ERRORBUF(R6),R1 :GET SIZE OF ERROR LOG BUFFER IN BYTES
0000 354 JSB ERL$CLOCMB :ALLOCATE ERROR MESSAGE BUFFER
0000 355 BLBC R0,30$ :IF LBC ALLOCATION FAILURE
0000 356 MOVL R2,UCB$C_EMB(R5) :SAVE ADDRESS OF ERROR MESSAGE BUFFER
0000 357 BLSW #UCB$C_ERLOGIP,UCB$C_STS(R5) :SIGNAL ERROR LOGGING IN PROGRESS
0000 358 MOVW 5*4(SPT,EMB$C_DV_ENTRY(R5)) :INSERT ENTRY TYPE
0000 359 ADDL #EMB$C_DV_CLASS,R2 :POINT TO DEVICE CLASS
0000 360
0000 361 ASSUME EMB$C_DV_TYPE EQ EMB$C_DV_CLASS+1
0000 362 MOVW UCB$C_DEVCLASS(R5),(R2)+ :INSERT DEVICE CLASS AND TYPE
0000 363 MOVL UCB$C_IRP(R5),R1 :GET ADDRESS OF I/O PACKET
0000 364
0000 365 ASSUME EMB$C_DV_RQPID EQ EMB$C_DV_TYPE+1
0000 366 MOVL IRP$C_PID(R1),(R2)+ :INSERT REQUESTER PROCESS ID
0000 367
0000 368 ASSUME EMB$C_DV_BOFF EQ EMB$C_DV_RQPID+4
0000 369 ASSUME EMB$C_DV_BCNT EQ EMB$C_DV_BOFF+2
0000 370 MOVL IRP$C_BOFF(R1),(R2)+ :INSERT TRANSFER PARAMETERS
0000 371
```

01	DD	0000	341
05	11	0002	342
		0004	343
7E	0060	8F	3C 0004 344
03	38	A5	E0 0009 345
		0081	31 000E 346
F7	009A	C5	E0 0011 347
		0082	C5 B6 0017 348
72	64	A5	E0 0018 349
		004F	8F BB 0020 350
53	28	A5	D0 0024 351
56	0088	C5	D0 0028 352
		51	16 A6 3C 002D 353
	00000251	EF	16 0031 354
		54	50 E9 0037 355
0094	C5	52	D0 003A 356
	64	A5	A8 003F 357
04	A2	14	B0 0043 358
		52	1C C0 0048 359
			004B 360
			004B 361
82	40	A5	B0 004B 362
51	58	A5	D0 004F 363
			0053 364
			0053 365
82	0C	A1	D0 0053 366
			0057 367
			0057 368
			0057 369
82	30	A1	D0 0057 370
			005B 371

82	00BC	C5	D0	005B	372	ASSUME	EMB\$\$_DV_MEDIA	EQ	EMB\$\$_DV_BCNT+2
				005B	373	MOVL	UCB\$\$_MEDIA(R5),	(R2)+	;INSERT SIZE OF DISK
				0060	374				
82	54	A5	B0	0060	375	ASSUME	EMB\$\$_DV_UNIT	EQ	EMB\$\$_DV_MEDIA+4
				0060	376	MOVW	UCB\$\$_UNIT(R5),	(R2)+	;INSERT UNIT NUMBER
				0064	377				
82	0082	C5	B0	0064	378	ASSUME	EMB\$\$_DV_ERRCNT	EQ	EMB\$\$_DV_UNIT+2
				0064	379	MOVW	UCB\$\$_ERRCNT(R5),	(R2)+	;INSERT NUMBER OF DEVICE ERRORS
				0069	380				
82	70	A5	D0	0069	381	ASSUME	EMB\$\$_DV_OPCNT	EQ	EMB\$\$_DV_ERRCNT+2
				0069	382	MOVL	UCB\$\$_OPCNT(R5),	(R2)+	;INSERT OPERATIONS COMPLETED
				006D	383				
50	1C	A5	D0	006D	384	ASSUME	EMB\$\$_DV_OWNUIC	EQ	EMB\$\$_DV_OPCNT+4
				006D	385	MOVL	UCB\$\$_ORB(R5),	R0	;GET ORB ADDRESS
82	60	D0		0071	386	MOVL	ORB\$\$_OWNER(R0),	(R2)+	;INSERT VOLUME OWNER UIC
				0074	387				
82	38	A5	D0	0074	388	ASSUME	EMB\$\$_DV_CHAR	EQ	EMB\$\$_DV_OWNUIC+4
				0074	389	MOVL	UCB\$\$_DEVCHAR(R5),	(R2)+	;INSERT DEVICE CHARACTERISTICS
				0078	390				
82	0090	C5	9B	0078	391	ASSUME	EMB\$\$_DV_SLAVE	EQ	EMB\$\$_DV_CHAR+4
				0078	392	MOVZBW	UCB\$\$_SLAVE(R5),	(R2)+	;INSERT SLAVE UNIT NUMBER
				007D	393				
82	20	A1	B0	007D	394	ASSUME	EMB\$\$_DV_FUNC	EQ	EMB\$\$_DV_SLAVE+2
				007D	395	MOVW	IRP\$\$_FUNC(R1),	(R2)+	;INSERT FUNCTION VALUE
				0081	396				
7E	52	10	C1	0081	397	ASSUME	EMB\$\$_DV_NAME	EQ	EMB\$\$_DV_FUNC+2
				0081	398	ADDL3	#EMB\$\$_DV_REGSAR-EMB\$\$_DV_NAME,R2,	-(SP)	;CALCULATE ADDRESS OF REGIST
	0269	30		0085	399	BSBW	ERL\$GETFULNAME		;Copy full device name
	50	8ED0		0088	400	POPL	R0		;Restore address of register dump area
	10	B6	16	008B	401	JSB	@DDT\$\$_REGDUMP(R6)		;CALL REGISTER DUMP ROUTINE
	004F	8F	BA	008E	402	POPR	#^M<R0,R1,R2,R3,R6>		;RESTORE REGISTERS
	5E	04	C0	0092	403	ADDL2	#4,SP		;REMOVE ENTRY TYPE FROM STACK
			05	0095	404	RSB			
				0096	405	.DSABL	LSB		

```
0096 407 .SBTTL LOG ASYNCHRONOUS DEVICE ATTENTIONS
0096 408
0096 409
0096 410 :+ ERL$DEVICEATTN - Log asynchronous device attention interrupts that are
0096 411 : not related to the current I/O operation that may be in progress.
0096 412
0096 413 INPUTS:
0096 414
0096 415 R5 => UCB
0096 416
0096 417 OUTPUTS:
0096 418
0096 419 If error logging is enabled for the device, an error log buffer
0096 420 is allocated, filled in and released. There may be an error log
0096 421 in progress for the current device, but this is not taken into
0096 422 account since the current attention interrupt is not related to
0096 423 the I/O that may be in progress.
0096 424
0096 425 :-
0096 426
0096 427 ERL$DEVICEATTN::
0096 428
56 004F 8F BB 0096 429 PUSH R0,R1,R2,R3,R6 ; Save registers.
51 0088 C5 D0 009A 430 MOVL UCBSL_DDT(R5),R6 ; Get address of DDT.
16 A6 3C 009F 431 MOVZWL DDT$W_ERRORBUF(R6),R1 ; R1=size of error log buffer in bytes.
0082 C5 B6 00A3 432 INCW UCBSW_ERRCNT(R5) ; Increment number of device errors.
16 E1 00A7 433 BBC #DEV$V_ELQ,-
5D 38 A5 00A9 434 UCBSL_DEVCHAR(R5),30$ ; If clr, error log disabled.
01A2 30 00AC 435 BSBW ERL$A[LOCEMB ; Allocate error message buffer.
57 50 E9 00AF 436 BLBC R0,30$ ; If LBC allocation failure.
52 DD 00B2 437 PUSHL R2 ; Save address of allocated buffer.
0062 8F B0 00B4 438 MOVW #EMBSB_DA,-
04 A2 00B8 439 EMBSW_DV_ENTRY(R2) ; Insert entry type.
64 A5 B0 00BA 440 MOVW UCBSW_STS(R5),- ; Save device status in buffer.
1A A2 00BD 441 EMBSW_DV_STS(R2)
12 A2 7C 00BF 442 CLRQ EMBSQ_DV_IOSB(R2) ; Clear irrelevant field.
52 1C C0 00C2 443 ADDL #EMBSB_DV_CLASS,R2 ; R2 => device class field.
00C5 444
82 40 A5 B0 00C5 446 ASSUME EMBSB_DV_TYPE EQ EMBSB_DV_CLASS+1
00C5 447 MOVW UCBSB_DEVCLASS(R5),(R2)+ ; Insert device class and type.
00C9 448
00C9 449 ASSUME EMBSL_DV_RQPID EQ EMBSB_DV_TYPE+1
00C9 450 ASSUME EMBSW_DV_BOFF EQ EMBSL_DV_RQPID+4
00C9 451 ASSUME EMBSW_DV_BCNT EQ EMBSW_DV_BOFF+2
82 7C 00C9 452 CLRQ (R2)+ ; Clear PID, BOFF and BCNT.
00CB 453
00CB 454 ASSUME EMBSL_DV_MEDIA EQ EMBSW_DV_BCNT+2
82 00BC C5 D0 00CB 455 MOVL UCBSL_MEDIA(R5),(R2)+ ; Insert size of disk.
00D0 456
00D0 457 ASSUME EMBSW_DV_UNIT EQ EMBSL_DV_MEDIA+4
82 54 A5 B0 00D0 458 MOVW UCBSW_UNIT(R5),(R2)+ ; Insert unit number.
00D4 459
00D4 460 ASSUME EMBSW_DV_ERRCNT EQ EMBSW_DV_UNIT+2
82 0082 C5 B0 00D4 461 MOVW UCBSW_ERRCNT(R5),(R2)+ ; Insert number of device errors.
00D9 462
00D9 463 ASSUME EMBSL_DV_OPCNT EQ EMBSW_DV_ERRCNT+2
```

82	70	A5	D0	00D9	464	MOVL	UCBSL_OPCNT(R5),(R2)+	; Insert operations completed.
				00DD	465			
				00DD	466	ASSUME	EMBSL_DV_OWNUIC EQ	EMBSL_DV_OPCNT+4
50	1C	A5	D0	00DD	467	MOVL	UCBSL_ORB(R5),R0	;GET ORB ADDRESS
	82	60	D0	00E1	468	MOVL	ORBSL_OWNER(R0),(R2)+	; Insert volume owner uic.
				00E4	469			
				00E4	470	ASSUME	EMBSL_DV_CHAR EQ	EMBSL_DV_OWNUIC+4
82	38	A5	D0	00E4	471	MOVL	UCBSL_DEVCHAR(R5),(R2)+	; Insert device characteristics.
				00E8	472			
				00E8	473	ASSUME	EMBSB_DV_SLAVE EQ	EMBSL_DV_CHAR+4
82	0090	C5	9B	00E8	474	MOVZBW	UCBSB_SLAVE(R5),(R2)+	; Insert slave unit number.
				00ED	475			
				00ED	476	ASSUME	EMBSW_DV_FUNC EQ	EMBSB_DV_SLAVE+2
	B2	B4		00ED	477	CLRW	(R2)+	; Clear Irrelevant function value.
				00EF	478			
				00EF	479	ASSUME	EMBST_DV_NAME EQ	EMBSW_DV_FUNC+2
7E	52	10	C1	00EF	480	ADDL3	#EMBSL_DV_REGSAY-EMBST_DV_NAME,R2,-(SP)	;CALCULATE ADDRESS OF REGIST
53	28	A5	D0	00F3	481	MOVL	UCBSL_DDB(R5),R3	; Get address of DDB
	01F7	30		00F7	482	BSBW	ERLSGETFULLNAME	; Copy full device name
	50	BED0		00FA	483	POPL	R0	; Restore address of register dump area
51	10	A6	D0	00FD	484	MOVL	DDTSL_REGDUMP(R6),R1	; R1 => register dump routine.
	61	16		0101	485	JSB	(R1)	; Call register dump routine.
	52	BED0		0103	486	POPL	R2	; Restore address of allocated buffer.
	021C	30		0106	487	BSBW	ERLSRELEASEMB	; Release this error log buffer.
	004F	8F	BA	0109	488	POPR	#^M<R0,R1,R2,R3,R6>	; Restore registers.
			05	010D	489	RSB		
				010E	490	.DSABL	LSB	

30\$:

```
010E 492 .SBTTL LOG SOFTWARE STATUS
010E 493
010E 494
010E 495 ERL$LOGSTATUS - Log software status corresponding to a logged message.
010E 496
010E 497 INPUTS:
010E 498 R0-R1 contain final I/O status
010E 499 R2 => MSCP end message
010E 500 R3 => UCB
010E 501 R5 => CDRP
010E 502
010E 503 OUTPUTS:
010E 504 An error log message (format EMBSPDEF) is allocated and filled in.
010E 505 All registers are preserved.
010E 506
010E 507
010E 508 ERL$LOGSTATUS::
010E 509
0082 C3 B6 010E 510 INCW UCBSW_ERRCNT(R3) ; Increment number of device errors.
71 38 A3 E1 0112 511 BBC #DEVSQ_ELQ,- ; If clear, error log disabled.
0114 512 UCBSL_DEVCHAR(R3),20$
0117 513
7E 50 7D 0117 514 MOVQ R0,-(SP) ; Save R0, R1, R2.
52 DD 011A 515 PUSHL R2
51 0050 8F 3C 011C 516 MOVZWL #EMBSK_SP_LENGTH,R1 ; R1 contains length of buffer to alloc
012D 30 0121 517 BSBW ERL$ALOC_EMB ; Allocate error message buffer.
5B 50 E9 0124 518 BLBC R0,10$ ; LBC implies allocation failure.
0127 519
0063 8F B0 0127 520 MOVW #EMBSK_SP,- ; Indicate type of error log buffer.
04 A2 012B 521 EMBW_SP_ENTRY(R2)
50 10 A2 9E 012D 522 MOVAB EMB$B_SP_CLASS(R2),R0 ; R0 => where to begin filling.
0131 523
0131 524 ASSUME UCBSB_DEVTYPE EQ UCBSB_DEVCLASS+1
0131 525 ASSUME EMB$B_SP_TYPE EQ EMB$B_SP_CLASS+1
80 40 A3 B0 0131 526 MOVW UCBSB_DEVCLASS(R3),(R0)+ ; Move Device type and class.
0135 527
0135 528 ASSUME EMB$W_SP_BOFF EQ EMB$B_SP_TYPE+1
80 D0 A5 B0 0135 529 MOVW CDRP$B_BOFF(R5),(R0)+ ; Copy BOFF.
0139 530
0139 531 ASSUME EMB$L_SP_BCNT EQ EMB$W_SP_BOFF+2
80 D2 A5 D0 0139 532 MOVL CDRP$C_BCNT(R5),(R0)+ ; Also byte count.
013D 533
013D 534 ASSUME EMB$L_SP_MEDIA EQ EMB$L_SP_BCNT+4
80 DB A5 D0 013D 535 MOVL CDRP$C_MEDIA(R5),(R0)+ ; Move media address (LBN).
0141 536
0141 537 ASSUME EMB$L_SP_RQPID EQ EMB$L_SP_MEDIA+4
80 AC A5 D0 0141 538 MOVL CDRP$C_PID(R5),(R0)+ ; Copy requesting PID.
0145 539
0145 540 ASSUME EMB$Q_SP_IOSB EQ EMB$L_SP_RQPID+4
80 04 AE 7D 0145 541 MOVQ 4(SP),(R0)+ ; Copy saved I/O status to buffer.
0149 542
0149 543 ASSUME EMB$W_SP_FUNC EQ EMB$Q_SP_IOSB+8
80 C0 A5 B0 0149 544 MOVW CDRP$Q_FUNC(R5),(R0)+ ; Copy I/O function code.
014D 545
014D 546 ASSUME EMB$W_SP_UNIT EQ EMB$W_SP_FUNC+2
80 54 A3 B0 014D 547 MOVW UCBSW_UNIT(R3),(R0)+ ; Copy unit number.
0151 548
```

80	70	A3	D0	0151	549	ASSUME	EMBSL_SP OPCNT EQ	EMBSW_SP UNIT+2
				0151	550	MOVL	UCBSL_OPENT(R3),(R0)+	; Copy cumulative operation count.
				0155	551			
80	0082	C3	B0	0155	552	ASSUME	EMBSW_SP ERRCNT EQ	EMBSL_SP OPCNT+4
				0155	553	MOVW	UCBSW_ERRCNT(R3),(R0)+	; And also cumulative error count.
				015A	554			
80	64	A3	B0	015A	555	ASSUME	EMBSW_SP UCBSTS EQ	EMBSW_SP ERRCNT+2
				015A	556	MOVW	UCBSW_STS(R3),(R0)+	; Copy UCB STS field.
				015E	557			
51	1C	A3	D0	015E	558	ASSUME	EMBSL_SP OWNUIC EQ	EMBSW_SP UCBSTS+2
	80	61	D0	015E	559	MOVL	UCBSL_ORB(R3),R1	; GET ORB ADDRESS
				0162	560	MOVL	ORBSL_OWNER(R1),(R0)+	; Copy device owner UIC.
				0165	561			
80	38	A3	D0	0165	562	ASSUME	EMBSL_SP CHAR EQ	EMBSL_SP OWNUIC+4
				0165	563	MOVL	UCBSL_DEVCHAR(R3),(R0)+	; Copy device characteristics.
				0169	564			
51	6E	D0	D0	0169	565	ASSUME	EMBSL_SP_CMDREF EQ	EMBSL_SP CHAR+4
	80	61	D0	0169	566	MOVL	(SP),R1	; R1 => MSCP end message.
				016C	567	MOVL	MSCP\$\$_CMD_REF(R1),(R0)+	; Copy command reference number (RSPID).
				016F	568			
				016F	569	ASSUME	EMBSL_SP_DEVNAM EQ	EMBSL_SP_CMDREF+4
7E	52	7D	016F	570	MOVQ	R2,-(SP)	; Save UCB & buffer base address(R2,R3)	
52	50	D0	0172	571	MOVL	R0,R2	; Get buffer address	
53	28	A3	D0	0175	572	MOVL	UCBSL_DDB(R3),R3	; Get DDB address
	0175	30	0179	573	BSBW	ERL\$GETFULLNAME	; Copy full device name	
52	8E	7D	017C	574	MOVQ	(SP)+,R2	; Restore R2 and R3	
			017F	575				
	01A3	30	017F	576	BSBW	ERL\$RELEASEMB	; Release filled in error buffer.	
			0182	577				
	52	8ED0	0182	578	POPL	R2	; Restore registers R2, R1, R0.	
50	8E	7D	0185	579	MOVQ	(SP)+,R0		
			0188	580				
		05	0188	581	RSB		; Return to caller.	

```
0189 583 .SBTTL LOG DRIVER MESSAGE
0189 584
0189 585
0189 586 ERL$LOGMESSAGE - Subroutine to allocate a message buffer, fill in a
0189 587 standard header, and then copy caller specified text to the rest
0189 588 of the buffer.
0189 589
0189 590 INPUTS:
0189 591 R0 = Code specifying message sub type.
0189 592 R1 = Length of caller specified text
0189 593 R2 => caller text
0189 594 R3 => UCB
0189 595
0189 596 OUTPUTS:
0189 597 Message allocated and filled. All registers preserved.
0189 598
0189 599
0189 600 ERL$LOGMESSAGE::
0189 601
0082 C3 B6 0189 602 INCW UCBSW_ERRCNT(R3) ; Increment total number of errors.
16 E1 018D 603 BBC #DEV$V_ELG,- ; Clear means error logging inhibited.
55 3B A3 018F 604 UCBSL_DEVCHAR(R3),20$
0192 605
7E 50 7D 0192 606 MOVQ R0,-(SP) ; Save registers R0-R5.
7E 52 7D 0195 607 MOVQ R2,-(SP)
7E 54 7D 0198 608 MOVQ R4,-(SP)
51 26 C0 019B 609 ADDL #EMBSK_LM_LENGTH,R1 ; Add message header to caller's length.
00B0 30 019E 610 BSBW ERL$ALCOEMB ; Allocate buffer.
3A 50 E9 01A1 611 BLBC R0,10$ ; LBC means allocation failure.
01A4 612
0064 52 DD 01A4 613 PUSHL R2 ; Save address of buffer.
8F B0 01A6 614 MOVW #EMBSK_LM,- ; Indicate type of error log buffer.
04 A2 01AA 615 EMBSW_LM_ENTRY(R2)
01AC 616
01AC 617 ASSUME UCBSB_DEVTYPE EQ UCBSB_DEVCLASS+1
01AC 618 ASSUME EMBSB_LM_TYPE EQ EMBSB_LM_CLASS+1
40 A3 B0 01AC 619 MOVW UCBSB_DEVCLASS(R3),- ; Begin to fill in buffer. Copy
10 A2 01AF 620 EMBSB_LM_CLASS(R2) ; Device type and class.
01B1 621
54 A3 B0 01B1 622 MOVW UCBSW_UNIT(R3),- ; Also copy device unit number.
12 A2 01B4 623 EMBSW_LM_UNIT(R2)
01B6 624
52 53 DD 01B6 625 PUSHL R3 ; Save UCB
53 14 A2 DE 01B8 626 MOVAL EMBSK_LM_DEVNAM(R2),R2 ; Get buffer address for device name
53 28 A3 DO 01BC 627 MCVL UCBSL_DDB(R3),R3 ; Get DDB address
012E 30 01C0 628 BSBW ERL$GETFULLNAME ; Copy full device name
53 53 B0 01C3 629 POPL R3 ; Restore UCB
52 6E DO 01C6 630 MOVL (SP),R2 ; Restore buffer base address
01C9 631
24 A2 14 AE B0 01C9 632 MOVW 20(SP),EMBSW_LM_MSGTYP(R2) ; Copy message subtype.
51 0C AE DO 01CE 633 MOVL 12(SP),R1 ; R1 => caller's text.
26 A2 61 18 AE 28 01D2 634 MOVCL 24(SP),(R1),EMBSW_LM_MSGTYP+2(R2) ; Copy caller's text.
52 B0 01D8 635 POPL R2 ; R2 => allocated buffer.
0147 30 01DA 636 BSBW ERL$RELEASEMB ; Release buffer.
01DE 637 10$:
54 8E 7D 01DE 638 MOVQ (SP)+,R4 ; Restore Registers R0-R5.
52 8E 7D 01E1 639 MOVQ (SP)+,R2
```

ERRORLOG
V04-000

- ERROR LOG SUPPORT ROUTINES
LOG DRIVER MESSAGE

K 5

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 16
(1)

50	8E	7D	01E4	640		MOVQ	(SP)+,R0
			01E7	641	20\$:		
		05	01E7	642		RSB	

; Return to caller.

```
01E8 644 .SBTTL ERL$LOG_DMSCP and ERL$LOG_TMSCP
01E8 645
01E8 646
01E8 647 :+
01E8 648 : Routines that respectively log invalid Disk and Tape MSCP messages.
01E8 649 :
01E8 650 : Inputs:
01E8 651 : R0 = type of message
01E8 652 : R1 = length of message
01E8 653 : R2 => message
01E8 654 : R3 => CDDB
01E8 655 :
01E8 656 : Outputs:
01E8 657 : All registers preserved.
01E8 658 :
01E8 659 : We want to log the following items in addition to the message
01E8 660 : and its type:
01E8 661 : 1. CDDBSB_SYSTEMID (6 bytes)
01E8 662 : 2. The ASCII string 'DISK' (4 bytes) or 'TAPE' (4 bytes)
01E8 663 : 3. CDDBSQ_CNTRLID (8 bytes)
01E8 664 :
01E8 665 :
01E8 666 : .enabl lsb
01E8 667
01E8 668 ERL$LOG_TMSCP::
01E8 669
01E8 670 PUSH R0,R1,R2,R3,R4,R5 ; Save registers.
01EA 671 MOVL #A/TAPE/,R4 ; R4 has string 'TAPE'.
01F1 672 BRB 10$ ; Branch around to common code.
01F3 673 ERL$LOG_DMSCP::
01F3 674
01F3 675 PUSH R0,R1,R2,R3,R4,R5 ; Save registers.
01F5 676 MOVL #A/DISK/,R4 ; R4 has string 'DISK'.
01FC 677 10$:
01FC 678 ADDL #<2+4+6+8 - ; R1 has length which is bumped by
01FF 679 +EMBSK_HD_LENGTH>,R1 ; 2 for the type, 4 for 'DISK' or
01FF 680 ; 'TAPE', 6 for SYSTEMID, 8 for
01FF 681 ; CNTRLID, and errorlog entry header
01FF 682 ; size.
004F 30 BSBW ERL$ALLOCEMB ; Allocate Errorlog Buffer.
27 50 E9 R0,20$ ; LBC means no allocate.
52 DD R2 ; Save R2=>Buffer.
0065 8F B0 MOVW #EMBSK_LOGMSCP,- ; Copy message class to buffer header.
04 A2 020B EMB$W_RD_ENTRY(R2)
52 10 C0 ADDL #EMBSK_HD_LENGTH,R2 ; R2 => beyond header.
82 04 AE B0 MOVW 4(SP),R2+ ; Copy message type (from saved regs).
82 54 D0 MOVW R4,(R2)+ ; Copy Class driver type.
82 20 A3 7D MOVQ CDDBSQ_CNTRLID(R3),(R2)+ ; Controller identifier.
82 0C A3 7D MOVQ CDDBSB_SYSTEMID(R3),(R2)+ ; And System ID.
08 AE 28 MOVCL 8(SP),- ; Get length from saved registers.
0C BE 0222 @12(SP),- ; also source address.
FE A2 0224 -2(R2) ; Target is -2 since SYSTEMID is 6 bytes.
52 8ED0 0226 R2 ; Restore R2=>Buffer.
00F9 30 0229 BSBW ERL$RELEASEMB ; Free Errorlog buffer.
3F BA 022C 20$:
05 05 022E POPR #M<R0,R1,R2,R3,R4,R5> ; Restore registers.
RSB
```

ERRORLOG
V04-000

- ERROR LOG SUPPORT ROUTINES
ERL\$LOG_DMSCP and ERL\$LOG_TMSCP

M 5

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 18
(1)

022F 701 .dsabl lsb

```

022F 703      .SBTTL  BUILD STARTUP AND POWERFAIL MESSAGES
022F 704      :+
022F 705      ERL$COLDSTART - LOG COLDSTART (SYSTEMBOOT)
022F 706
022F 707      THIS ROUTINE IS CALLED BY SYSINIT AFTER CORRECTLY SETTING THE SYSTEM
022F 708      TIME TO LOG THE BOOTING OF THE SYSTEM.
022F 709
022F 710      ERL$WARMSTART - LOG WARMSTART (POWER RECOVERY)
022F 711
022F 712      THIS ROUTINE IS CALLED BY POWERFAIL AFTER CORRECTING THE SYSTEM TIME
022F 713      TO LOG THE POWER FAIL AND RECOVERY.
022F 714
022F 715      INPUTS:
022F 716      NONE
022F 717
022F 718      OUTPUTS:
022F 719
022F 720      AN ERROR LOG BUFFER IS ALLOCATED AND FILLED WITH THE APPROPRIATE MESSAGE
022F 721      IF POSSIBLE AND THE ERROR LOG PROCESS AWAKENED IF NECESSARY.
022F 722      -
022F 723      .ENABL  LSB
022F 724      ERL$COLDSTART::
022F 725      MOVZWL  #EMB$C_CS,R3      :SET TYPE OF MESSAGE TO COLDSTART
0232 726      BRb    10$              :
0234 727      ERL$WARMSTART::
0234 728      MOVZWL  #EMB$C_WS,R3      :SET TYPE OF MESSAGE TO WARMSTART
0237 729      10$:  MOVZWL  #EMB$C_SU_LENGTH,R1  :SET SIZE OF MESSAGE TO ALLOCATE
023A 730      BSBB    ERL$ALCOEMB      :ALLOCATE AN ERROR LOG BUFFER
023C 731      BLBC    R0,20$           :BRANCH IF NONE AVAILABLE
023F 732      JSB     EXE$READ_TODR    :GET TIME TO LOG
0245 733      MOVL    R0,EMB$SU_DAYTIM(R2) :LOG TIME OF DAY CLOCK
0249 734      MOVW    R3,EMB$SU_ENTRY(R2) :SET MESSAGE TYPE
024D 735      BSBW    ERL$RELEASEMB      :RELEASE BUFFER
0250 736      20$:  RSB
0251 737      .DSABL  LSB

```

```

53 20 3C
03 11
53 24 3C
51 14 3C
15 10
11 50 E9
00000000 EF 16
10 A2 50 D0
04 A2 53 B0
00D5 30
05

```

```
0251 739 .SBTTL ALLOCATE ERROR MESSAGE BUFFER
0251 740
0251 741 * ERL$ALLOCEMB - ALLOCATE ERROR MESSAGE BUFFER
0251 742
0251 743 THIS ROUTINE IS CALLED TO ALLOCATE AN ERROR LOG MESSAGE BUFFER AND
0251 744 INITIALIZE ITS HEADER.
0251 745
0251 746 INPUTS:
0251 747
0251 748 R1 = SIZE OF MESSAGE BUFFER REQUIRED IN BYTES.
0251 749
0251 750 OUTPUTS:
0251 751
0251 752 R0 LOW BIT CLEAR INDICATES AN ALLOCATION FAILURE.
0251 753
0251 754 R0 LOW BIT SET INDICATES SUCCESSFUL ALLOCATION WITH:
0251 755
0251 756 R1 = ERROR SEQUENCE NUMBER.
0251 757 R2 = ADDRESS OF ALLOCATED ERROR MESSAGE BUFFER.
0251 758
0251 759 IN EITHER CASE THE UNIVERSAL ERROR SEQUENCE NUMBER IS INCREMENTED
0251 760 AND STORED IN THE BUFFER AT THE STANDARD PLACE, ALONG WITH THE TIME.
0251 761 AND THE ERROR LOG PROCESS MAY BE AWAKENED IF AN ERROR ALLOCATION
0251 762 BUFFER IS FOUND TO BE FULL.
0251 763
0251 764 R3 IS PRESERVED ACROSS CALL.
0251 765 :-
0251 766
0251 767 ERL$ALLOCEMB::
0251 768 DSBINT ;ALLOCATE ERROR MESSAGE BUFFER
0251 769 ADDL ;DISABLE ALL INTERRUPTS
0251 770 MOVZBL #EMBSK_LENGTH,R1 ;Add in size of header for message
0251 771 MOV ERL$GB_BUFIND,R0 ;GET CURRENT ALLOCATION BUFFER INDICATOR
0251 772 BBS #ERL$V_LOCK,ERL$B_FLAGS(R0),15$ ;GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
0251 773 10$: MOVL ERL$NEXT(R0),R2 ;GET ADDRESS OF NEXT AVAILABLE SPACE
0251 774 ADDL3 R1,R2,ERL$NEXT(R0) ;CALCULATE ADDRESS OF NEXT AVAILABLE SPACE
0251 775 CMPL ERL$END(R0),ERL$NEXT(R0) ;ENTRY FIT WITHIN BUFFER?
0251 776 BGEQU 20$ ;IF GEQU YES
0251 777 BISB #ERL$M_TIMER,ERL$GB_BUFIND ;SET TIMER ACTIVE
0251 778 MOVB #1,ERL$GB_BUFTIM ;FORCE ERROR LOG PROCESS WAKE
0251 779 15$: MOVL ERL$END(R0),ERL$NEXT(R0) ;INDICATE THAT BUFFER IS FULL
0251 780 XORB #1,ERL$GB_BUFIND ;SWITCH TO ALTERNATE BUFFER
0251 781 MOVZBL ERL$GB_BUFIND,R0 ;GET NEW BUFFER INDICATOR
0251 782 MOVL ERL$AL_BUFADDR[R0],R0 ;GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
0251 783 BBS #ERL$V_LOCK,ERL$B_FLAGS(R0),17$ ;IF SET, BUFFER BEING COPIED
0251 784 ADDL3 R1,ERL$NEXT(R0),R2 ;CALCULATE ADDRESS OF NEXT AVAILABLE SPACE
0251 785 CMPL ERL$END(R0),R2 ;ENTRY FIT WITHIN BUFFER?
0251 786 BGEQU 10$ ;IF GEQU YES
0251 787 17$: MOVL ERL$END(R0),ERL$NEXT(R0) ;INDICATE THAT BUFFER IS FULL
0251 788 CLRL R0 ;INDICATE ALLOCATION FAILURE
0251 789 BRB 30$
0251 790 20$: ADDL #EMBSK_LENGTH,R2 ;Point past the message header
0251 791 MFPR #PR$_SID,EMBSL_HD_SID(R2) ;Set system ID into message
0251 792 MOVZWL R1,EMBSW_SIZE(R2) ;Set size in message header
0251 793 MOVB ERL$B_BUFIND(R0),EMBSB_BUFIND(R2) ;SET RESPECTIVE BUFFER INDICATOR
0251 794 INCB ERL$B_BUSY(R0) ;INCREMENT MESSAGE BUSY COUNT
0251 795 MOVL ERL$GC_SEQUENCE,R1 ;GET CURRENT ERROR SEQUENCE NUMBER
```

50	00000408'EF	04	C0	0257	769	DSBINT
50	00000400'EF40	00	9A	025A	770	ADDL
	1E 03 A0	00	D0	0261	771	MOVZBL
	52 04 A0	00	E0	0269	772	MOVL
04 A0	52 51	00	D0	026E	773	BBS
04 A0	08 A0	00	C1	0272	774	10\$: MOVL
	42	00	D1	0277	775	ADDL3
00000409'EF	02	88	027C	776	CMPL	
00000408'EF	01	1E	027E	777	ERL\$END(R0),ERL\$NEXT(R0)	
04 A0	08 A0	88	0285	778	BGEQU	
00000408'EF	01	90	028C	779	20\$	
50	00000408'EF	01	0291	780	;IF GEQU YES	
50	00000400'EF40	00	0298	781	BISB	
0B 03 A0	00	9A	029F	782	#ERL\$M_TIMER,ERL\$GB_BUFIND	
52 04 A0	51	D0	02A7	783	;SET TIMER ACTIVE	
52 08 A0	00	E0	02AC	784	MOVZBL	
04 A0	08 A0	C1	02B1	785	ERL\$GB_BUFIND,R0	
	08 A0	D1	02B5	786	;GET NEW BUFFER INDICATOR	
	27	1E	02B7	787	MOVL	
	50	D0	02BC	788	ERL\$AL_BUFADDR[R0],R0	
	27	11	02BE	789	;GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR	
52 04	00	C0	02C0	790	BBS	
62 3E	00	DB	02C3	791	#ERL\$V_LOCK,ERL\$B_FLAGS(R0),17\$	
FC A2	51	3C	02C6	792	;IF SET, BUFFER BEING COPIED	
FE A2	02 A0	90	02CA	793	ADDL3	
	60	96	02CF	794	R1,ERL\$NEXT(R0),R2	
51	00000410'EF	D0	02D1	795	;CALCULATE ADDRESS OF NEXT AVAILABLE SPACE	

06 A2	00000000'EF	7D	02D8	796	MOVQ	EXESGQ	SYTIME,EMBSQ	DV	TIME(R2)	:	INSERT	CURRENT	TIME		
	0E A2 51	80	02E0	797	MOVW	R1,EMBSW	DV	ERRSEQ(R2)	-	:	INSERT	ERROR	SEQUENCE	NUMBER	
	50 01	D0	02E4	798	MOVL	#1,R0				:	SET	SUCCESS	INDICATOR		
	00000410'EF	D6	02E7	799	INCL	ERL\$GL_SEQUENCE				:	INCREMENT	UNIVERSAL	ERROR	SEQUENCE	NUMBER
			02ED	800	ENBINT					:	ENABLE	INTERRUPTS			
		05	02F0	801	RSB					:					

```
02F1 803 .SBTTL GET FULL DEVICE NAME
02F1 804 :+
02F1 805 : ERL$GETFULLNAME - GET FULL DEVICE NAME
02F1 806 :
02F1 807 : THIS ROUTINE IS CALLED TO COPY THE FULL DEVICE NAME (NODE NAME + DEVICE NAME)
02F1 808 : TO THE ERROR LOG BUFFER.
02F1 809 :
02F1 810 : INPUTS:
02F1 811 :
02F1 812 : R3 = address of DDR
02F1 813 : R2 = address of error log buffer
02F1 814 :
02F1 815 : OUTPUTS:
02F1 816 :
02F1 817 : If a node name exist in the system block, it is copied with the
02F1 818 : device name to the error log buffer.
02F1 819 :
02F1 820 : R0, R1, AND R3 ARE DESTROYED ACROSS CALL.
02F1 821 : -
02F1 822 :
02F1 823 ERL$GETFULLNAME::
51 14 A3 9E 02F1 824 MOVAB DDB$T_NAME(R3),R1 ; Get address of device name.
7E 81 9A 02F5 825 MOVZBL (R1)+,-(SP) ; Save the string length
53 34 A3 D0 02F8 826 MOVL DDB$L_SB(R3),R3 ; Get address of system block
1A 13 02FC 827 BEQL 20$ ; If EQL, go to move device name
53 44 A3 9E 02FE 828 MOVAB SB$T_NODENAME(R3),R3 ; Get address of nodename
50 83 9A 0302 829 MOVZBL (R3)+,R0 ; Get nodename length
11 13 0305 830 BEQL 20$ ; If eql, go move device name
62 6E 50 81 0307 831 ADDB3 R0,(SP),(R2) ; Nodename length + device name
82 82 96 030B 832 INCB (R2)+ ; Total string len. + 1 for '$'
82 83 90 030D 833 10$: MOVAB (R3)+,(R2)+ ; Copy nodename
FA 50 F5 0310 834 SOBGTR R0,10$
82 24 90 0313 835 MOVAB #^A/$/, (R2)+ ; Insert the '$'
03 11 0316 836 BRB 30$ ; Go move device name
82 6E 90 0318 837 20$: MOVAB (SP),(R2)+ ; Move dev. name len. to buffer
50 8E D0 031B 838 30$: MOVL (SP)+,R0 ; Get dev. name length
82 81 90 031E 839 40$: MOVAB (R1)+,(R2)+ ; Move device name into buffer
FA 50 F5 0321 840 SOBGTR R0,40$
05 0324 841 RSB ; Return to caller
0325 842
```

```
0325 844 .SBTTL RELEASE ERROR MESSAGE BUFFER
0325 845 :+
0325 846 : ERL$RELEASEMB - RELEASE ERROR MESSAGE BUFFER
0325 847 :
0325 848 : THIS ROUTINE IS CALLED TO RELEASE AN ERROR MESSAGE BUFFER FOR PROCESSING
0325 849 : BY THE ERROR LOG PROCESS.
0325 850 :
0325 851 : INPUTS:
0325 852 :
0325 853 : R2 = ADDRESS OF ERROR MESSAGE BUFFER.
0325 854 :
0325 855 : OUTPUTS:
0325 856 :
0325 857 : THE COMPLETED ERROR MESSAGE COUNT IS INCREMENTED IN THE RESPECTIVE
0325 858 : ALLOCATION BUFFER HEADER, THE MESSAGE IS SET VALID, AND THE BUSY
0325 859 : MESSAGE COUNT IS DECREMENTED IN THE RESPECTIVE ALLOCATION BUFFER
0325 860 : HEADER.
0325 861 :
0325 862 : R3 IS PRESERVED ACROSS CALL.
0325 863 :-
0325 864
0325 865 ERL$RELEASEMB::
0325 866 INCB EMB$B_VALID(R2) :RELEASE ERROR MESSAGE BUFFER
0325 867 MOVZBL EMB$B_BUFIND(R2),R0 :SET MESSAGE BUFFER VALID
0325 868 MOVL ERL$AC_BUFADDR(R0),R0 :GET BUFFER INDICATOR OF ALLOCATION BUFFER
0325 869 ADABI #^XFF,ERL$B_BUSY(R0) :GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
0325 870 BBBS #ERL$V_TIMER,ERL$GB_BUFFLAG,10$ :ADJUST BUSY AND COMPLETED MESSAGE COUNT
0325 871 CMPB #MAXMSG,ERL$B_MSGCNT(R0) :MAXIMUM NUMBER OF MESSAGES EXCEEDED?
0325 872 BGTRU 10$ :IF GTRU NO
0325 873 MOVB #1,ERL$GB_BUFTIM :FORCE ERROR LOG PROCESS WAKE
0325 874 10$: RSB :
```

50	00000400	'EF	40	96	0325	866
60	00FF	8F	58	0334	867	
0D	00000409	'EF	01	E3	0339	870
	01	A0	0A	91	0341	871
			07	1A	0345	872
0000040B	'EF	01	90	0347	873	
			05	034E	874	10\$:

```

034F 876      .SBTTL WAKE ERROR LOG FORMAT PROCESS
034F 877      :+
034F 878      : ERL$WAKE - WAKE ERROR LOG FORMAT PROCESS
034F 879      :
034F 880      : THIS ROUTINE IS CALLED ONCE A SECOND WHEN THE ERROR BUFFER TIMER IS ACTIVE.
034F 881      :
034F 882      : INPUTS:
034F 883      :
034F 884      :     NONE.
034F 885      :
034F 886      : OUTPUTS:
034F 887      :
034F 888      :     THE ERROR BUFFER TIMER IS DECREMENTED AND IF THE RESULT IS ZERO THE
034F 889      :     ERROR LOG FORMAT PROCESS IS AWAKENED.
034F 890      : -
034F 891      :
034F 892      ERL$WAKE::
034F 893      DECB      ERL$GB_BUFTIM      ;WAKE ERROR LOG FORMAT PROCESS
0355 894      BNEQ      10$                ;DECREMENT TIMER
0357 895      BICB      #ERL$M_TIMER,ERL$GB_BUFFLAG ;CLEAR TIMER ACTIVE FLAG
035E 896      MOVB      #MAXTIM,ERL$GB_BUFTIM ;RESET TIMER VALUE
0365 897      MOVL      ERL$GL_ERLPID,R1    ;GET ERROR LOG PROCESS ID
036C 898      BSBW      SCH$WAKE           ;WAKE ERROR LOG PROCESS
036F 899      RSB      10$:
0370 900
0370 901      .END

```

0000040B'EF 97 034F 893
18 12 0355 894
00000409'EF 02 8A 0357 895
0000040B'EF 1E 90 035E 896
51 0000040C'EF D0 0365 897
FC91' 30 036C 898
05 036F 899
0370 900
0370 901

ERRORLOG
Symbol table

- ERROR LOG SUPPORT ROUTINES

G 6

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 25
(1)

ADP_HANDLER	0000003E	R	03	EMBSL_UI_CSR	=	00000014		
ADP_UNEXP	00000000	R	03	EMBSL_UI_TR	=	00000010		
BUFT	00000000	R	02	EMBSQ_DV_IOSB	=	00000012		
BUF2	00000200	R	03	EMBSQ_DV_TIME	=	00000006		
BUGS_UNXINTEXC	*****	X	03	EMBSQ_SP_IOSB	=	00000020		
CDDBSB_SYSTEMID	= 0000000C			EMBSL_DV_NAME	=	0000003E		
CDDBSQ_CNTRLID	= 00000020			EMBSL_LM_DEVNAM	=	00000014		
CDRPSL_BCNT	= FFFFFFFD2			EMBSL_SP_DEVNAM	=	00000040		
CDRPSL_MEDIA	= FFFFFFFD8			EMBSW_DV_BCNT	=	00000024		
CDRPSL_PID	= FFFFFFFAC			EMBSW_DV_BOFF	=	00000022		
CDRPSW_BOFF	= FFFFFFFD0			EMBSW_DV_ENTRY	=	00000004		
CDRPSW_FUNC	= FFFFFFFC0			EMBSW_DV_ERRCNT	=	0000002C		
CPU_UNEXP	000000B8	R	03	EMBSW_DV_ERRSEQ	=	0000000E		
DDBSL_SB	= 00000034			EMBSW_DV_FUNC	=	0000003C		
DDBST_NAME	= 00000014			EMBSW_DV_STS	=	0000001A		
DDTSL_REGDUMP	= 00000010			EMBSW_DV_UNIT	=	0000002A		
DDTSW_ERRORBUF	= 00000016			EMBSW_HD_ENTRY	=	00000004		
DEVSU_ELQ	= 00000016			EMBSW_LM_ENTRY	=	00000004		
EMBSB_BUFIND	= FFFFFFFFE			EMBSW_LM_MSGTYP	=	00000024		
EMBSB_DV_CLASS	= 0000001C			EMBSW_LM_UNIT	=	00000012		
EMBSB_DV_SLAVE	= 0000003A			EMBSW_SIZE	=	FFFFFFFFC		
EMBSB_DV_TYPE	= 0000001D			EMBSW_SP_BOFF	=	00000012		
EMBSB_LM_CLASS	= 00000010			EMBSW_SP_ENTRY	=	00000004		
EMBSB_LM_TYPE	= 00000011			EMBSW_SP_ERRCNT	=	00000030		
EMBSB_SP_CLASS	= 00000010			EMBSW_SP_FUNC	=	00000028		
EMBSB_SP_TYPE	= 00000011			EMBSW_SP_UCBSTS	=	00000032		
EMBSB_VACID	= FFFFFFFF			EMBSW_SP_UNIT	=	0000002A		
EMBSL_CS	= 00000020			EMBSW_SU_ENTRY	=	00000004		
EMBSL_DA	= 00000062			EMBSW_UI_ENTRY	=	00000004		
EMBSL_DE	= 00000001			ERL\$ALOCEMB	=	00000251	RG	04
EMBSL_DT	= 00000060			ERL\$AL_BUFADDR	=	00000400	RG	02
EMBSL_LM	= 00000064			ERL\$B_BUFIND	=	00000002		
EMBSL_LOGMSCP	= 00000065			ERL\$B_BUSY	=	00000000		
EMBSL_SP	= 00000063			ERL\$B_FLAGS	=	00000003		
EMBSL_SU_LENGTH	= 00000014			ERL\$B_MSGCNT	=	00000001		
EMBSL_UI	= 00000061			ERL\$COLDSTART	=	0000022F	RG	04
EMBSL_UI_LENGTH	= 00000018			ERL\$C_LENGTH	=	0000000C		
EMBSL_WS	= 00000024			ERL\$DEVICEATTN	=	00000096	RG	04
EMBSK_HD_LENGTH	= 00000010			ERL\$DEVICERR	=	00000000	RG	04
EMBSK_LENGTH	= 00000004			ERL\$DEVICTMO	=	00000004	RG	04
EMBSK_LM_LENGTH	= 00000026			ERL\$GB_BUFFLAG	=	00000409	RG	02
EMBSK_SP_LENGTH	= 00000050			ERL\$GB_BUFIND	=	00000408	RG	02
EMBSL_DV_CHAR	= 00000036			ERL\$GB_BUFPTR	=	0000040A	RG	02
EMBSL_DV_MEDIA	= 00000026			ERL\$GB_BUFTIM	=	0000040B	RG	02
EMBSL_DV_OPCNT	= 0000002E			ERL\$GETFULLNAME	=	000002F1	RG	04
EMBSL_DV_OWNUIC	= 00000032			ERL\$GL_ERLPID	=	0000040C	RG	02
EMBSL_DV_REGSAV	= 0000004E			ERL\$GL_SEQUENCE	=	00000410	RG	02
EMBSL_DV_RQPID	= 0000001E			ERL\$LOGMESSAGE	=	00000189	RG	04
EMBSL_HD_SID	= 00000000			ERL\$LOGSTATUS	=	0000010E	RG	04
EMBSL_SP_BCNT	= 00000014			ERL\$LOG_DMSCP	=	000001F3	RG	04
EMBSL_SP_CHAR	= 00000038			ERL\$LOG_TMSCP	=	000001E8	RG	04
EMBSL_SP_CMDREF	= 0000003C			ERL\$END	=	00000008		
EMBSL_SP_MEDIA	= 00000018			ERL\$NEXT	=	00000004		
EMBSL_SP_OPCNT	= 0000002C			ERL\$TIMER	=	00000002		
EMBSL_SP_OWNUIC	= 00000034			ERL\$RELEASEMB	=	00000325	RG	04
EMBSL_SP_RQPID	= 0000001C			ERL\$UNEXP	=	000000B8	RG	03
EMBSL_SU_DAYTIM	= 00000010			ERL\$VECO	=	000000B8	RG	03

ERRORLOG
Symbol table

- ERROR LOG SUPPORT ROUTINES

H 6

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 26
(1)

ERL\$VEC100	000000B8	RG	03	ERL\$VEC308	00000034	RG	03
ERL\$VEC104	000000B8	RG	03	ERL\$VEC312	00000038	RG	03
ERL\$VEC108	000000B8	RG	03	ERL\$VEC316	0000003C	RG	03
ERL\$VEC112	000000B8	RG	03	ERL\$VEC320	000000B8	RG	03
ERL\$VEC116	000000B8	RG	03	ERL\$VEC324	00000000	RG	03
ERL\$VEC120	000000B8	RG	03	ERL\$VEC328	00000004	RG	03
ERL\$VEC124	000000B8	RG	03	ERL\$VEC332	00000008	RG	03
ERL\$VEC128	000000B8	RG	03	ERL\$VEC336	0000000C	RG	03
ERL\$VEC132	000000B8	RG	03	ERL\$VEC340	00000010	RG	03
ERL\$VEC136	000000B8	RG	03	ERL\$VEC344	00000014	RG	03
ERL\$VEC140	000000B8	RG	03	ERL\$VEC348	00000018	RG	03
ERL\$VEC144	000000B8	RG	03	ERL\$VEC352	0000001C	RG	03
ERL\$VEC148	000000B8	RG	03	ERL\$VEC356	00000020	RG	03
ERL\$VEC152	000000B8	RG	03	ERL\$VEC360	00000024	RG	03
ERL\$VEC156	000000B8	RG	03	ERL\$VEC364	000000B8	RG	03
ERL\$VEC160	000000B8	RG	03	ERL\$VEC368	00000028	RG	03
ERL\$VEC164	000000B8	RG	03	ERL\$VEC372	0000002C	RG	03
ERL\$VEC168	000000B8	RG	03	ERL\$VEC376	00000030	RG	03
ERL\$VEC172	000000B8	RG	03	ERL\$VEC380	00000034	RG	03
ERL\$VEC176	000000B8	RG	03	ERL\$VEC384	00000038	RG	03
ERL\$VEC180	000000B8	RG	03	ERL\$VEC388	0000003C	RG	03
ERL\$VEC184	000000B8	RG	03	ERL\$VEC392	00000000	RG	03
ERL\$VEC188	000000B8	RG	03	ERL\$VEC396	00000004	RG	03
ERL\$VEC192	000000B8	RG	03	ERL\$VEC400	00000008	RG	03
ERL\$VEC196	000000B8	RG	03	ERL\$VEC404	0000000C	RG	03
ERL\$VEC200	000000B8	RG	03	ERL\$VEC408	000000B8	RG	03
ERL\$VEC204	000000B8	RG	03	ERL\$VEC412	00000000	RG	03
ERL\$VEC208	000000B8	RG	03	ERL\$VEC416	00000004	RG	03
ERL\$VEC212	000000B8	RG	03	ERL\$VEC420	00000008	RG	03
ERL\$VEC216	000000B8	RG	03	ERL\$VEC424	0000000C	RG	03
ERL\$VEC220	000000B8	RG	03	ERL\$VEC428	000000B8	RG	03
ERL\$VEC224	000000B8	RG	03	ERL\$VEC432	00000000	RG	03
ERL\$VEC228	000000B8	RG	03	ERL\$VEC436	00000004	RG	03
ERL\$VEC232	000000B8	RG	03	ERL\$VEC440	00000008	RG	03
ERL\$VEC236	000000B8	RG	03	ERL\$VEC444	0000000C	RG	03
ERL\$VEC240	000000B8	RG	03	ERL\$VEC448	000000B8	RG	03
ERL\$VEC244	000000B8	RG	03	ERL\$VEC452	00000038	RG	03
ERL\$VEC248	000000B8	RG	03	ERL\$VEC456	0000003C	RG	03
ERL\$VEC252	000000B8	RG	03	ERL\$VEC460	00000000	RG	03
ERL\$VEC256	00000000	RG	03	ERL\$VEC464	00000004	RG	03
ERL\$VEC260	00000004	RG	03	ERL\$VEC468	00000008	RG	03
ERL\$VEC264	00000008	RG	03	ERL\$VEC472	0000000C	RG	03
ERL\$VEC268	0000000C	RG	03	ERL\$VEC476	000000B8	RG	03
ERL\$VEC272	00000010	RG	03	ERL\$VEC480	00000000	RG	03
ERL\$VEC276	00000014	RG	03	ERL\$VEC484	00000004	RG	03
ERL\$VEC280	000000B8	RG	03	ERL\$VEC488	00000008	RG	03
ERL\$VEC284	00000018	RG	03	ERL\$VEC492	0000000C	RG	03
ERL\$VEC288	0000001C	RG	03	ERL\$VEC496	000000B8	RG	03
ERL\$VEC292	00000020	RG	03	ERL\$VEC500	00000000	RG	03
ERL\$VEC296	00000024	RG	03	ERL\$VEC504	00000004	RG	03
ERL\$VEC300	00000028	RG	03	ERL\$VEC508	00000008	RG	03
ERL\$VEC304	0000002C	RG	03		0000000C	RG	03
	00000030	RG	03		00000010	RG	03
					00000014	RG	03
					00000018	RG	03
					0000001C	RG	03
					00000020	RG	03
					00000024	RG	03
					00000028	RG	03
					0000002C	RG	03
					00000030	RG	03
					00000034	RG	03
					00000038	RG	03
					0000003C	RG	03

ERRORLOG
Symbol table

- ERROR LOG SUPPORT ROUTINES

I 6

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 27
(1)

ERLSVEC52	000000B8	RG	03	VECTOR	= 0000023C
ERLSVEC56	000000B8	RG	03	VNUM	= 00000100
ERLSVEC60	000000B8	RG	03		
ERLSVEC64	000000B8	RG	03		
ERLSVEC68	000000B8	RG	03		
ERLSVEC72	000000B8	RG	03		
ERLSVEC76	000000B8	RG	03		
ERLSVEC8	000000B8	RG	03		
ERLSVEC80	000000B8	RG	03		
ERLSVEC84	000000B8	RG	03		
ERLSVEC88	000000B8	RG	03		
ERLSVEC92	000000B8	RG	03		
ERLSVEC96	000000B8	RG	03		
ERLSVEC RETURN	000000C0	RG	03		
ERLSV_LOCK	= 00000000				
ERLSV_TIMER	= 00000001				
ERLSWAKE	0000034F	RG	04		
ERLSWARMSTART	00000234	RG	04		
EXESGL_CONFREG	*****	X	03		
EXESGL_SYSTIME	*****	X	04		
EXESMCHK_PRTCT	*****	X	03		
EXESREAD_TODR	*****	X	04		
IOSGL_SCB INTO	*****	X	03		
IOSV_INHERLOG	= 0000000B				
IRPSC_PID	= 0000000C				
IRPSW_BOFF	= 00000030				
IRPSW_FUNC	= 00000020				
MAXMSG	= 0000000A				
MAXTIM	= 0000001E				
MCHKSM_LOG	= 00000001				
MCHKSM_NEXM	= 00000004				
MMGSGL_SBICONF	*****	X	03		
MSCPSL_CMD_REF	= 00000000				
NDTS_DR32	= 00000030				
NEXUS	= 00000010				
ORBSL_OWNER	= 00000000				
PR\$ IPL	= 00000012				
PR\$ SID	= 0000003E				
SBST_NODENAME	= 00000044				
SCHSWAKE	*****	X	04		
UCBSB_DEVCLASS	= 00000040				
UCBSB_DEVTYPE	= 00000041				
UCBSB_SLAVE	= 00000090				
UCBSL_DDB	= 00000028				
UCBSL_DDT	= 00000088				
UCBSL_DEVCHAR	= 00000038				
UCBSL_EMB	= 00000094				
UCBSL_IRP	= 00000058				
UCBSL_MEDIA	= 000000BC				
UCBSL_OPCNT	= 00000070				
UCBSL_ORB	= 0000001C				
UCBSM_ERLOGIP	= 00000004				
UCBSV_ERLOGIP	= 00000002				
UCBSW_ERRCNT	= 00000082				
UCBSW_FUNC	= 0000009A				
UCBSW_STS	= 00000064				
UCBSW_UNIT	= 00000054				

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$260	00000414 (1044.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD
\$AEXENONPAGED	000000C7 (199.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
WIONONPAGED	00000370 (880.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.07	00:00:01.74
Command processing	108	00:00:00.49	00:00:04.75
Pass 1	549	00:00:23.65	00:01:09.35
Symbol table sort	0	00:00:03.40	00:00:11.63
Pass 2	174	00:00:04.51	00:00:14.71
Symbol table output	34	00:00:00.28	00:00:01.92
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	905	00:00:32.44	00:01:44.15

The working set limit was 1950 pages.
130939 bytes (256 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2271 non-local and 34 local symbols.
901 source lines were read in Pass 1, producing 25 object records in Pass 2.
41 pages of virtual memory were used to define 40 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	28
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	35

2304 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:ERRORLOG/OBJ=OBJ\$:ERRORLOG MSRC\$:ERRORLOG/UPDATE=(ENH\$:ERRORLOG)+EXECMLS/LIB

0374 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

